CS241 #22 – Pipes and file seeks

Write some example code that uses an unnamed pipe to send a message from the parent to the child.

What is fseek and ftell? How would you use them?

What happens to the other process if you fclose after forking?

What happens to the other process if you fseek before forking?

What happens to the other process if you fseek after forking?

Why does pwrite exist? When would you use it?

What is an named pipe and an unnamed pipe?

What signals can a pipe generate and when?

How would you modify your pipe code to send an integer value of a variable?

Why is it necessary to close the pipe's unused filedescriptors after forking?

How would you fix/improve this code?

|  |  |
| --- | --- |
| pthread\_mutex\_t m;  pthread\_cond\_t cv;  int in, out, count;  void\* buffer[16]  void enqueue(void\* ptr) {  p\_m\_lock(&m);  while(count < 16) {}  pthread\_mutex\_unlock(&m);  p\_cond\_broadcast(&cv);  count ++;  buffer[ (in++) % 16 ] = ptr;  } | void\* dequeue() {  p\_m\_lock(&m);  while(count == 0) {}  void\* result = buffer[ (out++) % 16 ];  p\_cond\_broadcast(&cv);  pthread\_mutex\_unlock(&m);  count --;  return result;  } |
| void pipe\_or\_quit(int\*result) {  if( 0 == pipe(result) ) return; else quit("pipe");  }  void create\_pipes(int\* array6) {  pipe\_or\_quit(array6);  pipe\_or\_quit(array6 +2);  pipe\_or\_quit(array6 +4);  }  void exec\_or\_quit(const char \*program, const char \*\*args, int old\_err\_fd) {  execv(program, (char\*const\*) args);  dup2(old\_err\_fd, 2);  quit("execv");  } | int run(const char \*testname, const char \*program, const char \*\*args,  const char \*input, char \*\*output,  char \*\*erroroutput, int \*waitresult) {  if (testname) printf("%s: Running %s\n", testname, program);  int pipes[6];  create\_pipes(pipes);  pid\_t childid = fork\_or\_quit();  if(childid ==0 ) { //Child should close 'in'(input), out(output) err(output)  // close unused end of pipes  close(pipes[1]); close(pipes[2]);close(pipes[4]);  int old\_err\_fd = dup(2);  dup2\_or\_quit(pipes[0] /\*read from \*/,0);  dup2\_or\_quit(pipes[3] /\*write to\*/, 1);  dup2\_or\_quit(pipes[5] /\*write to\*/ ,2);  alarm(ALARM\_TIMEOUT\_SECONDS);  exec\_or\_quit(program, args, old\_err\_fd);  } |